Project: Streamlining Security Across Environments with DevSecOps

PHASE 2- SOLUTION ARCHITECTURE

College Name: KNS Institute of Technology

Members:

- **Name : Mohammed Thayeeb Shariff [Team Lead]**
  USN : 1KN21IS031 [EMAIL]
  CAN ID : 32887773
- **Name : Nidith V S**
  USN : 1KN21IS038
  CAN ID : 32888579
- **Name : Ramachandragowda S Patil**
  USN : 1KN21CS082
  CAN ID : 32888557
- **Name : Sateesh Biradar**
  USN : 1KN21IS041
  CAN ID : 32889102

---

1. LOGICAL ARCHITECTURE

A. Frontend:

I. Developed using HTML, CSS, and JavaScript to create responsive and interactive user interfaces.

II. Provides users with features like sign-up/login, appointment booking, and a contact us form for complaints or special requests.

III. Synchronizes with the calendar system for doctor appointments and visualizes admin panel statistics through dynamic UI components.

B. Backend:

I. Built with Python and uses a Flask framework for handling business logic and exposing RESTful APIs.

II. Integrates static file management and file uploads for handling user and admin resources.

III. Implements role-based access control (RBAC) for secure access to administrative features.

IV. Provides APIs for managing appointments, doctor assignments, and statistical reporting.

SOLUTION ARCHITECTURE:

PROJECT STRUCTURE:

ORTHOSECURE/

├── app/
│ ├── __pycache__/
│ ├── static/
│ ├── templates/
│ ├── __init__.py
│ ├── database.py
│ ├── Dockerfile
│ ├── main.py
│ ├── requirements.txt
│ ├── wait-for.sh
├── db/
│ ├── Dockerfile
│ ├── init.sql
├── kubernetes/
├── tests/
├── .env
├── .gitlab-ci.yml
├── docker-compose.yml
├── LICENSE.txt
├── Readme.md

Kubernetes Structure (Separated)

kubernetes/

├── configmaps/
│ ├── app-config.yaml
│ ├── mysql-config.yaml
├── deployments/

```
│   ├── app-deployment.yaml
│   ├── dummy.yaml
│   ├── mysql-deployment.yaml
│   ├── phpmyadmin-deployment.yaml
│   ├── security-context.yaml
├── policies/
│   ├── admission-controllers/
│   │   ├── gatekeeper-constraints.yaml
│   ├── namespaces/
│   │   ├── secure-namespace.yaml
│   ├── rbac/
│   │   ├── role-restricted-deployer.yaml
│   │   ├── rolebinding-restricted.yaml
│   ├── network-policy.yaml
│   ├── pod-security-policy.yaml
├── pvc/
│   ├── mysql-pvc.yaml
├── secrets/
│   ├── mysql-secrets.yaml
├── service-accounts/
│   ├── app-service-account.yaml
├── services/
│   ├── app-service.yaml
│   ├── mysql-service.yaml
│   ├── phpmyadmin-service.yaml
├── tests/
│   ├── test-pod-security.yaml
│   ├── test-rbac-pod.yaml
├── kubernetes.txt
```

TECHNOLOGY STACK

1. **Development and Integration**
    - Languages and Frameworks: Python flask, HTML, CSS, JS
    - CI/CD Tools: GitHub Actions, GitLab CI/CD
    - Containerization and Orchestration: Docker, Kubernetes
2. **Security Tools**
    - SAST: SonarQube
    - DAST: OWASP ZAP
    - Dependency Scanning: Snyk
    - Container Security: Docker Scout
3. **Monitoring and Logging**
    - Application Monitoring: Prometheus, Grafana
4. **Compliance and Governance**
    - Compliance Tools: Checkmarx, Nessus
5. **Collaboration and Documentation**
    - Collaboration Platforms: Slack, Microsoft Teams
    - Documentation Tools: MS Word, Confluence
6. **Cloud and Infrastructure**
    - Cloud Providers: IBM Cloud [partially AWS]
    - IaC Tools: Terraform

This technology stack is purposefully designed to meet the demands of a proactive DevSecOps framework, addressing the challenges of speed, security, and multi-environment complexities.

PROJECT FILES AND DIRECTORY CREATION

To develop the DevSecOps project structure, we can use basic file system commands to create the directory and file structure. Below is a step-by-step guide using mkdir for directories and touch for files, assuming to use a Linux terminal or a similar environment (like WSL on Windows):

Step 1: Create main project directory and subdirectories

mkdir -p
ORTHOSECURE/{app/{static,templates,__pycache__},db,kubernetes/{configmaps,deployments,policies/{admission-controllers,namespaces,rbac},pvc,secrets,service-accounts,services,tests},tests}

Step 2: Create files in the app directory

touch ORTHOSECURE/app/{__init__.py,database.py,Dockerfile,main.py,requirements.txt,wait-for.sh}

Step 3: Create files in the db directory

touch ORTHOSECURE/db/{Dockerfile,init.sql}

Step 4: Create Kubernetes files

touch ORTHOSECURE/kubernetes/configmaps/{app-config.yaml,mysql-config.yaml}

touch ORTHOSECURE/kubernetes/deployments/{app-deployment.yaml,dummy.yaml,mysql-deployment.yaml,phpmyadmin-deployment.yaml,security-context.yaml}

touch ORTHOSECURE/kubernetes/policies/admission-controllers/gatekeeper-constraints.yaml

touch ORTHOSECURE/kubernetes/policies/namespaces/secure-namespace.yaml

touch ORTHOSECURE/kubernetes/policies/rbac/{role-restricted-deployer.yaml,rolebinding-restricted.yaml}

touch ORTHOSECURE/kubernetes/policies/{network-policy.yaml,pod-security-policy.yaml}

touch ORTHOSECURE/kubernetes/pvc/mysql-pvc.yaml

touch ORTHOSECURE/kubernetes/secrets/mysql-secrets.yaml

touch ORTHOSECURE/kubernetes/service-accounts/app-service-account.yaml

touch ORTHOSECURE/kubernetes/services/{app-service.yaml,mysql-service.yaml,phpmyadmin-service.yaml}

touch ORTHOSECURE/kubernetes/tests/{test-pod-security.yaml,test-rbac-pod.yaml}

touch ORTHOSECURE/kubernetes/kubernetes.txt

Step 5: Create root-level files

touch ORTHOSECURE/{.env,.gitlab-ci.yml,docker-compose.yml,LICENSE.txt,Readme.md}

Final Structure Validation -- >

To verify the structure:

tree DevSecOps_Project

VERSION CONTROL SETUP

To ensure that the development team is working collaboratively and tracking changes efficiently, we will set up a **GitHub repository** for version control.

Step 1: Install Git

Check if Git is installed:

git --version

Step 2: Set Up Git

Set your username and email:

git config --global user.name "Your Name"

git config --global user.email "your.email@example.com"

Step 3: Initialize a Repository

Navigate to your project directory:

cd orthosecure

git init

Step 4: Add Files to Version Control

Add all project files:

git add .

Step 5: Commit Files

Save your changes with a message:

git commit -m "Initial project setup"

Step 6: Link to a Remote Repository

Create a repository on GitHub, GitLab, or another platform. Then link it:

git remote add origin https://github.com/thay9211/orthosecure.git

---

Step 7: Push Changes

Push your code to the remote repository:

git branch -M main

git push -u origin main

FUTURE PLANS (GOAL-ORIENTED WITH STRATEGIES AND TECH STACK):

1. **Plan 1: Advanced Threat Intelligence**
    - **Goal**: Utilize AI/ML for predictive threat detection and proactive defense.
    - **Strategy**: Integrate AI/ML models into monitoring tools like Prometheus and ELK Stack to identify anomalies in real time.
    - **Tech Stack**: TensorFlow, PyTorch, Prometheus, ELK Stack.
2. **Plan 2: Security-as-Code**
    - **Goal**: Codify security policies for repeatable and scalable implementations.
    - **Strategy**: Create Infrastructure as Code (IaC) templates using tools like Terraform with embedded security standards.
    - **Tech Stack**: Terraform, AWS CloudFormation, HashiCorp Vault.
3. **Plan 3: Multi-Cloud Security Expansion**
    - **Goal**: Ensure consistent security across hybrid and multi-cloud environments.
    - **Strategy**: Leverage cloud-native security tools from AWS, Azure, and GCP to establish a unified multi-cloud security posture.
    - **Tech Stack**: Kubernetes, Istio, Cloud-native security tools (AWS Shield, Azure Security Center).
4. **Plan 4: Open-Source DevSecOps Framework**

- **Goal**: Foster innovation and broad adoption by releasing the framework as open-source.
- **Strategy**: Host the framework on GitHub, document it comprehensively, and build a community for contributions.
- **Tech Stack**: GitHub, Markdown for documentation, Swagger for API documentation.